**EXPERIMENT NO: -05**

**Aim**: - Write a program for adding element from a binary tree

Name: - Nishant Bhandari

Roll No: -06

Subject: -Fundamental Algorithm

Class: -SY-CS

Sign: -

**Source code: -**

class Node:

def \_\_init\_\_(self, key):

self.key = key

self.left = None

self.right = None

def inorder(root):

if root is not None:

inorder(root.left)

print (root.key)

inorder(root.right)

def insert( node, key):

if node is None:

return Node(key)

if key < node.key:

node.left = insert(node.left, key)

else:

node.right = insert(node.right, key)

return node

def minValueNode( node):

current = node

while(current.left is not None):

current = current.left

return current

def deleteNode(root, key):

if root is None:

return root

if key < root.key:

root.left = deleteNode(root.left, key)

elif(key > root.key):

root.right = deleteNode(root.right, key)

else:

if root.left is None :

temp = root.right

root = None

return temp

elif root.right is None :

temp = root.left

root = None

return temp

temp = minValueNode(root.right)

root.key = temp.key

root.right = deleteNode(root.right , temp.key)

return root

root = None

root = insert(root, 50)

root = insert(root, 30)

root = insert(root, 20)

root = insert(root, 40)

root = insert(root, 70)

root = insert(root, 60)

root = insert(root, 80)

print("Inorder traversal of the given tree")

inorder(root)

print ("\nDelete 20")

root = deleteNode(root, 20)

print ("Inorder traversal of the modified tree")

inorder(root)

print ("\nDelete 30")

root = deleteNode(root, 30)

print ("Inorder traversal of the modified tree")

inorder(root)

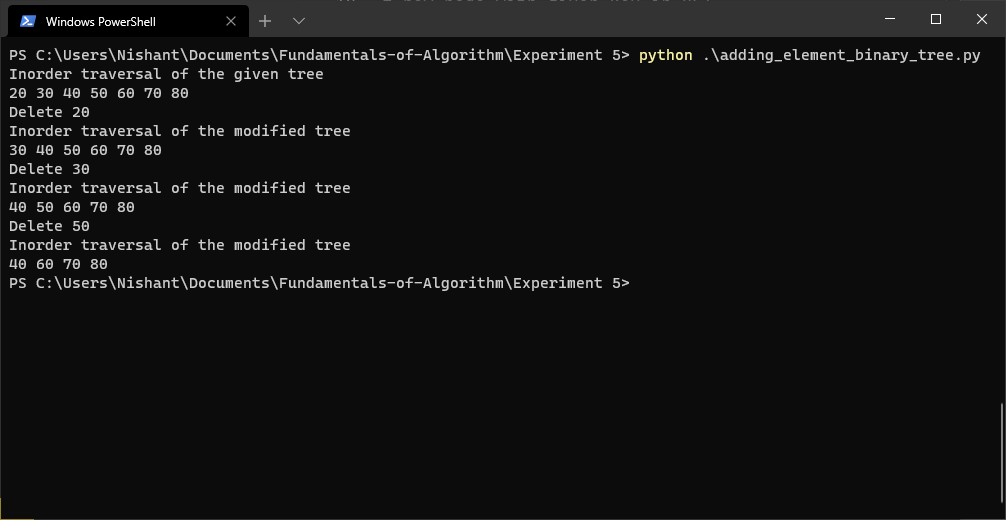
print ("\nDelete 50")

root = deleteNode(root, 50)

print ("Inorder traversal of the modified tree")

inorder(root)

**OUTPUT: -**

****

**Conclusion**: - thus we have studied about how to Implementing an element in Binary search tree.